Monadic Translation of Multi-Staged Languages

Ludovic Patey
ludovic.patey@ens.fr

Kwangkeun Yi
kwang@ropas.snu.ac.kr

ROPAS

December 23, 2011

Summary

Plan

Program Transformations

Goals

- Compilation
- Static analysis

Examples

- Unstaging Translation
- CPS Transformation
- SPS Transformation

Plan

CPS Transformation

Idea

Making continuations explicit by transforming each expression into a function taking its continuation as a parameter.

Source Language

$$e ::= i \mid x \mid \lambda x.e \mid \texttt{fix}\ f\ x.e \mid e\ e$$

Target Language

$$e ::= i \mid x \mid \lambda x.e \mid \texttt{fix}\ f\ x.e \mid e\ e$$

CPS Transformation

$$(\texttt{TCON}) \quad i \mapsto \lambda k.k\ i$$

$$(\texttt{TVAR}) \quad x \mapsto \lambda k.k\ x$$

$$(\texttt{TABS}) \quad \frac{e \mapsto \underline{e}}{\lambda x.e \mapsto \lambda k.k\ \lambda x.\underline{e}}$$

$$(\texttt{TFIX}) \quad \frac{e \mapsto \underline{e}}{\texttt{fix}\ f\ x.e \mapsto \lambda k\ k.\texttt{fix}\ f\ x.\underline{e}}$$

$$(\texttt{TAPP}) \quad \frac{e_1 \mapsto \underline{e_1} \qquad e_2 \mapsto \underline{e_2}}{e_1\ e_2 \mapsto \lambda k.\underline{e_1}\ \lambda m.\underline{e_2}\ \lambda n.m\ n\ k}$$

Semantics Preservation

$$
\begin{aligned}
\Psi(i) &= i \\
\Psi(x) &= x \\
\Psi(\lambda x.e) &= \lambda x.\underline{e} \\
\Psi(\texttt{fix}\ f\ x\ e) &= \texttt{fix}\ f\ x\ \underline{e}
\end{aligned}
$$

Semantics Preservation

Lemma (Value Translation)

$$v \mapsto \lambda k.k\ \Psi(v)$$

Lemma (Substitution Preservation)

$$[x \mapsto v]e \mapsto [x \mapsto \Psi(v)]\underline{e}$$

Lemma (Free Variables Preservation)

$$FV(e) = FV(\underline{e})$$

Semantics Preservation

Theorem (Natural Semantics Preservation)

$$\frac{e \Rightarrow v}{\underline{e} \Rightarrow \lambda k.k\ \Psi(v)}$$

Theorem (Type Translation)

$$\frac{e : \tau}{\underline{e} : (\underline{\tau} \to answer) \to answer}$$

$where\ \underline{\tau} = \tau\ \ and\ \underline{\tau_1 \to \tau_2} = \underline{\tau_1} \to (\underline{\tau_2} \to answer) \to answer$

Theorem (Simulation)

$$\underline{e}\ k \to^{\star} e : k \qquad \frac{e_1 \to e_2}{e_1 : k \to e_2 : k}$$

SPS Transformation

Idea

Internalizing memory management by transforming each expression into a function taking a store as a parameter.

Source Language

$$\begin{aligned} e ::= \ & i \mid x \mid \lambda x.e \mid \texttt{fix}\ f\ x.e \mid e\ e \\ & \mid l \mid \texttt{ref}\ e \mid !\ e \mid e := e \end{aligned}$$

Target Language

$$\begin{aligned} e ::= \ & i \mid x \mid \lambda x.e \mid \texttt{fix}\ f\ x.e \mid e\ e \\ & \mid \langle e, e \rangle \mid \texttt{match}\ e\ \texttt{with}\ \langle v, v \rangle \rightarrow e \\ & \mid s \mid l \mid \texttt{store\_alloc}\ e\ e \\ & \mid \texttt{store\_read}\ e\ e \mid \texttt{store\_write}\ e\ e\ e \end{aligned}$$

SPS Transformation

(TCON) $$i \mapsto \lambda s.\, \langle i, s \rangle$$

(TVAR) $$x \mapsto \lambda s.\, \langle x, s \rangle$$

(TABS) $$\frac{e \mapsto \underline{e}}{\lambda x.e \mapsto \lambda s.\, \langle \lambda x.\underline{e}, s \rangle}$$

(TFIX) $$\frac{e \mapsto \underline{e}}{\texttt{fix}\ f\ x.e \mapsto \lambda s.\, \langle \texttt{fix}\ f\ x.\underline{e}, s \rangle}$$

(TAPP) $$\frac{e_1 \mapsto \underline{e_1} \qquad e_2 \mapsto \underline{e_2})}{\begin{array}{l} e_1\ e_2 \mapsto \lambda s.\texttt{match}\ \underline{e_1}\ s\ \texttt{with}\ \langle v_{e_1}, s_1 \rangle \to \\ \qquad\qquad \texttt{match}\ \underline{e_2}\ s_1\ \texttt{with}\ \langle v_{e_2}, s_2 \rangle \to (v_{e_1}\ v_{e_2})\ s_2 \end{array}}$$

Semantics Preservation

Lemma (Value Translation)

$$v \mapsto \lambda s. \langle \Psi(v), s \rangle$$

Lemma (Substitution Preservation)

$$[x \mapsto v]e \mapsto [x \mapsto \Psi(v)]\underline{e}$$

Lemma (Free Variables Preservation)

$$FV(e) = FV(\underline{e})$$

Semantics Preservation

Theorem (Natural Semantics Preservation)

$$\frac{e \Rightarrow v}{\underline{e} \Rightarrow \lambda s.\, \langle \Psi(v), s \rangle}$$

Theorem (Type Translation)

$$\frac{e : \tau}{\underline{e} : store \rightarrow \underline{\tau} \times store}$$

where $\underline{\tau} = \tau$ *and* $\underline{\tau_1 \rightarrow \tau_2} = \underline{\tau_1} \rightarrow store \rightarrow \underline{\tau_2} \times store$

Theorem (Simulation)

$$\underline{e}\ s \rightarrow^{\star} e : s \qquad \frac{e_1 \rightarrow e_2}{e_1 : s \rightarrow e_2 : s}$$

Two main similarities

- In shape of the transformation
- In semantics preservation properties

Similarities

$$(\texttt{TCON}) \qquad i \mapsto \lambda k.k\ i \qquad\qquad (\texttt{TCON}) \qquad i \mapsto \lambda s.\,\langle i, s\rangle$$

$$(\texttt{TVAR}) \qquad x \mapsto \lambda k.k\ x \qquad\qquad (\texttt{TVAR}) \qquad x \mapsto \lambda s.\,\langle x, s\rangle$$

$$(\texttt{TABS}) \quad \frac{e \mapsto \underline{e}}{\lambda x.e \mapsto \lambda k.k\ \lambda x.\underline{e}} \qquad\qquad (\texttt{TABS}) \quad \frac{e \mapsto \underline{e}}{\lambda x.e \mapsto \lambda s.\,\langle \lambda x.\underline{e}, s\rangle}$$

Similarities

(TAPP) $$\frac{e_1 \mapsto \underline{e_1} \qquad e_2 \mapsto \underline{e_2}}{e_1\ e_2 \mapsto \lambda k.\underline{e_1}\ \lambda m.\underline{e_2}\ \lambda n.m\ n\ k}$$

(TAPP) $$\frac{e_1 \mapsto \underline{e_1} \qquad e_2 \mapsto \underline{e_2})}{\begin{array}{l} e_1\ e_2 \mapsto \lambda s.\texttt{match}\ \underline{e_1}\ s\ \texttt{with}\ \langle v_{e_1}, s_1\rangle \rightarrow \\ \qquad\qquad \texttt{match}\ \underline{e_2}\ s_1\ \texttt{with}\ \langle v_{e_2}, s_2\rangle \rightarrow (v_{e_1}\ v_{e_2})\ s_2 \end{array}}$$

Monadic Translation

Idea

Capturing the essence of previous translations by using abstract operators verifying algebraic properties.

Source Language

$$e ::= i \mid x \mid \lambda x.e \mid \texttt{fix}\ f\ x.e \mid e\ e$$

Target Language

$$e ::= ?$$

Monadic Translation

$$(\texttt{TCON}) \quad i \mapsto \text{RET } i$$

$$(\texttt{TVAR}) \quad x \mapsto \text{RET } x$$

$$(\texttt{TABS}) \quad \frac{e \mapsto \underline{e}}{\lambda x.e \mapsto \text{RET } \lambda x.\underline{e}}$$

$$(\texttt{TFIX}) \quad \frac{e \mapsto \underline{e}}{\texttt{fix } f \ x.e \mapsto \text{RET } \texttt{fix } f \ x.\underline{e}}$$

$$(\texttt{TAPP}) \quad \frac{e_1 \mapsto \underline{e_1} \qquad e_2 \mapsto \underline{e_2}}{e_1 \ e_2 \mapsto \text{BIND } \underline{e_1} \ \lambda m.\text{BIND } \underline{e_2} \ \lambda n.m \ n}$$

Algebraic Properties

Monadic operators must verify the following properties

- Local Preservation
  - $FV(\textsc{ret}\ v) = FV(v)$
  - $FV(\textsc{bind}\ e_1\ e_2) = FV(e_1) \cup FV(e_2)$
  - $[x \mapsto v_1]\textsc{ret}\ v_2 = \textsc{ret}\ [x \mapsto v_1]v_2$
  - $[x \mapsto v]\textsc{bind}\ e_1\ e_2 = \textsc{bind}\ [x \mapsto v]e_1\ [x \mapsto v]e_2$
- Contextual Equivalence
  - $(\lambda x.e)\ v \simeq [x \mapsto v]e$
  - $\textsc{bind}\ (\textsc{ret}\ v)\ (\lambda x.e) \simeq [x \mapsto v]e$
  - $\textsc{bind}\ e_1\ (\lambda x.e_2) \simeq \textsc{bind}\ e_1'\ (\lambda x.e_2)$ if $e_1 \simeq e_1'$

Semantics Preservation

Lemma (Value Translation)

$$v \mapsto \text{RET } \Psi(v)$$

Lemma (Substitution Preservation)

$$[x \mapsto v]e \mapsto [x \mapsto \Psi(v)]\underline{e}$$

Lemma (Free Variables Preservation)

$$FV(e) = FV(\underline{e})$$

Semantics Preservation

Theorem (Contextual Equivalence)

$$\frac{e \Rightarrow v}{\underline{e} \simeq \text{RET } \Psi(v)}$$

Theorem (Type Translation)

$$\frac{e : \tau}{\underline{e} : \underline{\tau}\ mon}$$

where $\underline{\tau} = \tau$ *and* $\underline{\tau_1 \rightarrow \tau_2} = \underline{\tau_1} \rightarrow (\underline{\tau_2}\ mon)$

Monadic Operators

- Continuation Monad
  - $\text{RET } v =_{\texttt{def}} \lambda k.k\ v$
  - $\text{BIND } e_1\ e_2 =_{\texttt{def}} \lambda k.(e_1\ e_2)\ k$
  - $e_1 \simeq e_2$ iff $\forall k \in Value.\ e_1\ k \xrightarrow{\star} v \xleftarrow{\star} e_2\ k.$
  - $\tau\ mon =_{\texttt{def}} (\tau \rightarrow answer) \rightarrow answer$
- State Monad
  - $\text{RET } v =_{\texttt{def}} \lambda s.\ \langle e, s \rangle$
  - $\text{BIND } e_1\ e_2 =_{\texttt{def}} \lambda s.\texttt{match}\ e_1\ \texttt{swith}\ \langle v_e, s' \rangle \rightarrow (e_2\ v_e)\ s'$
  - $e_1 \simeq e_2$ iff $\forall s \in Store.\ e_1\ s \xrightarrow{\star} v \xleftarrow{\star} e_2\ s.$
  - $\tau\ mon =_{\texttt{def}} store \rightarrow \tau \times store$

Advantages

- Helps to design a transformation
- Extracts the essence of the transformation
- Makes proofs much shorter
- And many other advantages...

Remark

Simulation theorem is missing, even if true for Continuation and State monad.

Plan

Language

Source Language

$$
\begin{aligned}
e ::= \;& i \mid x \mid \lambda x.e \mid e\ e \mid \texttt{fix}\ f\ x.e \\
& \mid \texttt{ref}\ e \mid !\ e \mid l \mid e := e \\
& \mid \texttt{box}\ e \mid \texttt{unbox}\ e \mid \texttt{lift}\ e \mid \texttt{run}\ e
\end{aligned}
$$

$$
\begin{aligned}
\texttt{run}\ (\texttt{box}\ e) &\rightarrow e \\
\texttt{unbox}\ (\texttt{box}\ e) &\rightarrow e \\
\texttt{lift}\ e &\stackrel{\star}{\rightarrow} \texttt{box}\ v
\end{aligned}
$$

Substitution

$$
\begin{aligned}
&[x, 0, \emptyset \mapsto_\rho v]\texttt{box}\ x\ (\lambda x.x\ (\texttt{unbox}\ x)) \\
&= \texttt{box}\ [x, 1, \emptyset \mapsto_\rho v](x\ \lambda x.x\ (\texttt{unbox}\ x)) \\
&= \texttt{box}\ [x, 1, \emptyset \mapsto_\rho v]x\ [x, 1, \emptyset \mapsto_\rho v](\lambda x.x\ (\texttt{unbox}\ x)) \\
&= \texttt{box}\ [x, 1, \emptyset \mapsto_\rho v]x\ (\lambda x.[x, 1, \{1\} \mapsto_\rho v](x\ (\texttt{unbox}\ x))) \\
&= \texttt{box}\ [x, 1, \emptyset \mapsto_\rho v]x\ (\lambda x.[x, 1, \{1\} \mapsto_\rho v]x\ [x, 1, \{1\} \mapsto_\rho v](\texttt{unbox}\ x))) \\
&= \texttt{box}\ [x, 1, \emptyset \mapsto_\rho v]x\ (\lambda x.[x, 1, \{1\} \mapsto_\rho v]x\ (\texttt{unbox}\ [x, 0, \emptyset \mapsto_\rho v]x)))
\end{aligned}
$$

$$
[x, n, S \mapsto_\rho v]y = \begin{cases} v & \text{if } x = y \text{ and } \rho^+(n, S) \\ y & \text{otherwise} \end{cases}
$$

Lattice of Replacement Predicates

Replacement predicates form a complete lattice using the pointwise partial order. $\mathcal{L}_{\mathbb{R}} = \left(\mathbb{R}, \dot{\leq}, \bot, \top, \cap_{\mathbb{R}}, \cup_{\mathbb{R}}\right)$ where

- $\bot(n, S) = 0$ for all $n \in \mathbb{N}$ and $S \subseteq \mathbb{N}$.
- $\top(n, S) = 1$ for all $n \in \mathbb{N}$ and $S \subseteq \mathbb{N}$.
- $(\cap_{\mathbb{R}} R)(n, S) = min\,\{\rho(n, S) \mid \rho \in R\}$ for all $R \subseteq \mathbb{R}$
- $(\cup_{\mathbb{R}} R)(n, S) = max\,\{\rho(n, S) \mid \rho \in R\}$ for all $R \subseteq \mathbb{R}$

Lattice of Substitutions

Lattice $\mathcal{L}_{\mathbb{R}}$ induces a complete lattice over staged substitutions $\mathcal{L}_{\mapsto} = \left(\{\mapsto_\rho\}_{\rho\in\mathbb{R}}, \leq, \mapsto_\bot, \mapsto_\top, \cap, \cup\right)$ where

- $\mapsto_{\rho_1} \leq \mapsto_{\rho_2}$ iff $\rho_1 \dot{\leq} \rho_2$.
- $\cap\{\mapsto_\rho \mid \rho \in S\} = \mapsto_{\cap_{\mathbb{R}} S}$.
- $\cup\{\mapsto_\rho \mid \rho \in S\} = \mapsto_{\cup_{\mathbb{R}} S}$.

$\mapsto_\bot$ coincides with substitution of lisp-like multi-staged calculus.

CPS Transformation

Definitions

$$\begin{array}{ll} \mathit{Context} & \kappa ::= \lambda k.e\ \lambda h.[\cdot]\ k \mid \lambda k.e\ \lambda h.\kappa\ k \\ \mathit{Context\ Stack} & K ::= \bot \mid K, \kappa \end{array}$$

Transformation

(TRUN)
$$\frac{e \mapsto (\underline{e}, K)}{\texttt{run}\ e \mapsto (\lambda k.\underline{e}\ \lambda m.\texttt{run}\ m\ k, K)}$$

(TBOX)
$$\frac{e \mapsto (\underline{e}, (K, \kappa))}{\texttt{box}\ e \mapsto (\kappa[\texttt{box}\ \underline{e}], K)} \qquad \frac{e \mapsto (\underline{e}, \bot)}{\texttt{box}\ e \mapsto (\lambda k.k\ \texttt{box}\ \underline{e}, \bot)}$$

(TUNB)
$$\frac{e \mapsto (\underline{e}, K)}{\texttt{unbox}\ e \mapsto (\texttt{unbox}\ h, (K, \lambda k.\underline{e}\ (\lambda h.[\cdot]\ k))))}$$

(TLIF)
$$\frac{e \mapsto (\underline{e}, K)}{\texttt{lift}\ e \mapsto (\lambda k.\underline{e}\ \lambda m.\texttt{lift}\ m\ k, K)}$$

SPS Transformation

Definitions

$$\begin{array}{ll} \textit{Context} & \kappa ::= \lambda s.\mathtt{match}\ e\ s\ \mathtt{with}\ \langle v_e, s' \rangle \to (\lambda h.[\cdot])\ v_e\ s' \\ & \quad\ \mid \lambda s.\mathtt{match}\ e\ s\ \mathtt{with}\ \langle v_e, s' \rangle \to (\lambda h.\kappa)\ v_e\ s' \\ \textit{Context Stack} & K ::= \bot \mid K, \kappa \end{array}$$

Transformation

$$\frac{e \mapsto (\underline{e}, K)}{\mathtt{run}\ e \mapsto (\lambda s.\mathtt{match}\ \underline{e}\ s\ \mathtt{with}\ \langle v_e, s' \rangle \to (\mathtt{run}\ v_1)\ s', K)}$$

$$\frac{e \mapsto (\underline{e}, (K, \kappa))}{\mathtt{box}\ e \mapsto \kappa[\mathtt{box}\ \underline{e}], K)} \qquad \frac{e \mapsto (\underline{e}, \bot)}{\mathtt{box}\ e \mapsto (\lambda s.\ \langle (\mathtt{box}\ \underline{e}), s \rangle, \bot)}$$

$$\frac{e \mapsto (\underline{e}, K)}{\mathtt{unbox}\ e \mapsto (\mathtt{unbox}\ h, (K, \lambda s.\mathtt{match}\ \underline{e}\ s\ \mathtt{with}\ \langle v_e, s' \rangle \to (\lambda h.[\cdot])\ v_e\ s'))}$$

$$\frac{e \mapsto (\underline{e}, K)}{\mathtt{lift}\ e \mapsto (\lambda s.\mathtt{match}\ \underline{e}\ s\ \mathtt{with}\ \langle v_e, s' \rangle \to (\mathtt{lift}\ v_1)\ s', K)}$$

Monadic Translation

Definitions

$$\begin{array}{ll} \mathit{Context} & \kappa ::= \text{BIND } e\ \lambda h.[\cdot] \mid \text{BIND } e\ \lambda h.\kappa \\ \mathit{Context\ Stack} & K ::= \bot \mid K, \kappa \end{array}$$

Translation

(TRUN) $$\frac{e \mapsto (\underline{e}, K)}{\texttt{run } e \mapsto (\text{BIND } \underline{e}\ (\lambda h.\texttt{run } h), K)}$$

(TBOX) $$\frac{e \mapsto (\underline{e}, (K, \kappa))}{\texttt{box } e \mapsto (\kappa[\texttt{box } \underline{e}], K)} \quad \frac{e \mapsto (\underline{e}, \bot)}{\texttt{box } e \mapsto (\text{RET } (\texttt{box } \underline{e}), \bot)}$$

(TUNB) $$\frac{e \mapsto (\underline{e}, K)}{\texttt{unbox } e \mapsto (\texttt{unbox } h, (K, \text{BIND } \underline{e}\ (\lambda h.[\cdot])))}$$

(TLIF) $$\frac{e \mapsto (\underline{e}, K)}{\texttt{lift } e \mapsto (\text{BIND } \underline{e}\ (\lambda h.\texttt{lift } h), K)}$$

Semantics Preservation

Lemma (Value Preservation)

$$\frac{v \in Value^0 \qquad depth(v) = 0}{v \mapsto (\text{RET } \Psi(v), \bot)}$$

Lemma (Substitution Preservation)

$$\frac{v \in Value^0 \qquad v \mapsto (\underline{v}, \bot) \qquad e \mapsto (\underline{e}, \bot)}{[x, 0, \emptyset \mapsto_\rho v]e \mapsto ([x, 0, \emptyset \mapsto_\rho \Psi(v)]\underline{e}, \bot)}$$

Lemma (Free Variables Preservation)

$$\frac{e \mapsto (\underline{e}, \bot)}{FV_\rho(e, 0, B_\bot) = FV_\rho(\underline{e}, 0, B_\bot)}$$

Semantics Preservation

Theorem (Contextual Equivalence)

$$\frac{e \mapsto (\underline{e}, \bot) \qquad M_{\emptyset}, e \xrightarrow{0^{\star}} v, M \qquad v \in Value^0}{\underline{e} \overset{0}{\simeq} \text{RET } \Psi(v)}$$

Theorem (Type Translation)

$$\frac{e : \tau}{\underline{e} : \underline{\tau}\ mon}$$

where $\underline{\tau} = \tau$ *and* $\underline{\tau_1 \to \tau_2} = \underline{\tau_1} \to (\underline{\tau_2}\ mon)$

Plan

Conclusion

My goals:

- Extend monadic translation to multi-staged calculi.
- Give a stronger notion of monad to recover simulation property
- Side effects on monadic operators

References

W. Choi, B. Aktemur, K. Yi, and M. Tatsuta.
Static analysis of multi-staged programs via unstaging translation.
ACM SIGPLAN Notices, 46(1):81–92, 2011.

X. Leroy.
Functional programming and type systems.
http://gallium.inria.fr/~xleroy/mpri/2-4/.

E. Moggi.
Computational lambda-calculus and monads.
In *Logic in Computer Science, 1989. LICS'89, Proceedings., Fourth Annual Symposium on*, pages 14–23. IEEE, 1989.

L. Patey and K. Yi.
Cps transformation of lisp-like multi-staged languages.
2011.